a) whether the core words of the two elements are in the same class; and

b) where the two elements are both chunks whether both chunks have the same phrasetag (as seen in Table 2).

A match in both cases might, for example, be given a score of 2, a score of 1 being given for a match in one case, and a score of 0 being given otherwise.

In order to calculate the degree of syntactic match between words in the elements, the program controls the computer to find to what level of the hierarchical classification the corresponding words in the elements are syntactically similar. A match of word categories might be given a score of 5, a match of sub-classes a score of 2 and a match of classes a score of 1. For example, if the reference sentence has [VG is_VBZ argued_VVN VG] and the input sentence has [VG was_VBDZ beaten_VVN VG] then 'is_VBZ' only matches 'was_VBDZ' to the extent that both are classified as verbs. Therefore a score of 1 would be given on the basis of the first word. With regard to the second word, 'beaten_VVN' and 'argued_VVN' fall into identical word categories and hence would be given a score of 5. The two scores are then added to give a total score of 6.

The third component of each element similarity measure is the negative magnitude of the difference in the number of words in the reference element, e_r, and the number of words in the element of the input sentence, e_i. For example, if an element of the input sentence has one word and an element of the reference sequence has three words, then the third component is -2.

A weighted addition is then performed on the three components to yield an element similarity measure (match(e_i,e_r) in the above pseudo-code).

Those skilled in the art will thus appreciate that the table calculation step 102 results in the generation of a table giving element similarity measures between every element in the corpus 52 and every element in the input sentence.

Then, in step 103, a subject element counter (m) is initialised to 1. The value of the counter indicates which of the elements of the input sentence is currently subject to a determination of whether it is to be followed by a boundary. Thereafter, the program controls the computer to execute an outermost loop of instructions (steps 104 to 125) repeatedly. Each iteration of the outermost loop of instructions corresponds to a consideration of a different subject element of the input sentence. It will be seen that each execution of the final instruction (step 125) in the outermost loop results in the next iteration of the outermost loop looking at the element in the input sentence which immediately follows the input sentence element considered in the previous iteration. Step 124 ensures that the outermost loop of instructions ends once the last element in the input sentence has been considered.

The outermost loop of instructions (steps 104 to 125) begins with the setting of a best match value to zero (step 104). Also, a current reference element count (e_r) is initialised to 1 (step 106).

Within the outermost loop of instructions (steps 104 to 125), the program controls the computer to repeat some or all of an intermediate loop of instructions (steps 108 to 121) as many times as there are elements in the prosodic structure corpus 52. Each iteration of the intermediate loop of instructions (steps 108 to 121) therefore corresponds to a particular subject element in the input sentence (determined by the current iteration of the outermost loop) and a particular reference element in the corpus 52 (determined by the current iteration of the intermediate loop). Steps 120 and 121 ensure that the intermediate loop of instructions (steps 108 to 121) is carried out for every element in the corpus 52 and ends once the final element in the corpus has been considered.

The intermediate loop of instructions (steps 108 to 121) starts by defining (step 108) a search sequence around the subject element of the input sentence.

The start and end of the search sequence are given by the expressions:

srch_seq_start = min(1, m – no_of_elements_before)
srch_seq_end = max(no_of_input_sentence_elements, m + no_of_elements_after)

In the preferred embodiment, no_of_elements_before is chosen to be 10, and no_of_elements_after is chosen to be 4. It will be realised that the search sequence therefore includes the current element m, up to 10 elements before it and up to 4 elements after it.

In step 110 a sequence similarity measure is reset to zero. In step 112 a measure of the similarity between the search sequence and a sequence of reference elements is calculated. The reference sequence has the current reference element (i.e. that set in the previous execution of step 121) as it core element. The reference sequence contains this core element as well as the four elements that precede it and the ten elements that follow it (i.e. the reference sequence is of the same length as the search sequence). The calculation of the sequence similarity measure involves carrying out first and second innermost loops of instructions. Pseudo-code for the first innermost loop of instructions is given below:

FOR current_position_in_srch_seq (=p) = srch_seq_start to srch_seq_end
s.s.m = s.s.m + weight(p) * match(srch_element_p, corres_ref_element)
NEXT

Where s.s.m is an abbreviation for sequence similarity measure.

In carrying out the steps represented by the above pseudo-code, in effect, the subject element of the input sentence (set in step 103 or 125) is aligned with the core reference element. Once those elements are aligned, the element similarity measure between each element of the search sequence and the corresponding element in the reference sequence is found. A weighted addition of those element similarity measures is then carried out to obtain a first component of a sequence similarity measure. The measures of the degree of matching are found in the values obtained in step 102. The weight applied to each of the constituent element